1 Struts2 极速表单验证框架使用说明书

Struts+	**struts-plus** *Struts enhancement, includes quick form validation, permission mgmt. Struts 框架增强，包括极速表单验证，权限管理等.*

子模块之 Struts 2 极速表单验证框架

作者: 刘长炯(beansoft@126.com)
日期: 2010-01-20
版本: 0.1

修订历史

2010-01-20 初始版

1 Struts2 极速表单验证框架使用说明书 1
2 Struts 2 极速表单验证框架简介 2
3 项目许可 3
4 下载本项目 3
5 演示应用 4
6 整合步骤step by step 8
7 验证规则表达式详解 12
8 和RapidValidation配合实现前后台一致验证 13
9 如何扩展验证规则 14
10 下一版本开发计划 17

2 Struts 2 极速表单验证框架简介

Struts Plus, Struts + 项目, 计划包含极速表单验证框架, 权限管理等子模块. 目前主要关注于 Struts 2 方面的生产力提高项目, 主要基于注解方式进行增强, 并提供抽象层, 使项目仅仅依赖于 Struts 本身, 而不需要强制您使用某种业务层和数据层框架. 项目的许可采用 MIT 协议, 您可自由的使用本项目, 但需遵循 MIT 协议保留本人版权签名及 MIT 协议本身.

本文主要讲述子模块 Struts 2 极速表单验证框架(Struts 2 QuickFormValidation).
本验证机制基于表达式和注解(Annotation), 验证规则的扩展通过实现验证接口进行.

诞生历史:

2009 年 12 月 18 日晚提出框架原理, 并于 2009 年 12 月 19 日晨提出实现思路, 最终框架核心初步完成于 2009 年 12 月 20 日(周日), 2010-01-20 日整理发布.

在项目的开发中, 表单验证需要用去开发者大量的时间, 尤其是为了保证前后台的验证规则一致时, 更是需要大量的人力物力. 虽然 Struts 2 本身的验证框架可以保证前后台一致, 但在使用和整合方式上, 需要编写大量的 Validation XML 文件, 费力费时容易出错, 尤其是实际开发中需要反复调整各种验证规则及其组合, 这时就更加困难; 扩展 Struts 2 的验证框架更是需要比较繁琐的步骤. 针对这些挑战, 笔者试图提供一种简便易用的方式来进行表单验证, 即基于 Java 注解和验证规则表达式.
如下面的代码即可实现如下的验证要求: 1) 用户名不能为空, 最小长度 5, 最大长度 10, 只能输入英文字母; 2) 年龄字段只能输入数字, 最小值为 18, 最大值为 100; 验证失败时, 转向 input 页面并显示出错信息: ${validationErrors}.

```
    @FormValidator(input="form_tag", value="user.name,用户名,required
min-length-5 max-length-10 validate-alpha;" +
            "age,年龄,validate-digits min-value-18 max-value-100;")
    public String form_tag() throws Exception {
        return SUCCESS;
    }
```

- 最快速的后台表单验证框架
- 简洁,快速的验证语法
- 无需编写验证提示信息
- 错误消息在指定地方显示
- 支持组合验证
- 易于扩展
- 基于标准的 Java Annotation 添加验证,易于和现有项目的无缝集成
- 特殊应用场景支持,如:密码确认,结束日期必须大于开始日期

- 支持前后台一致的验证规则

本框架主要是为了和前台基于JavaScript的开源验证框架RapidValidation(http://code.google.com/p/rapid-validation/)使用统一的验证规则，最大限度的减少重复的后台验证代码，使用方式简便，扩展方便.

本框架依赖于:

- Struts 2
- Java 5 或者更高版本

3 项目许可

The MIT License

Copyright (c) 2010 刘长炯(beansoft@126.com)

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

4 下载本项目

您可访问 http://code.google.com/p/struts-plus/downloads/list 来获取最新源代码包

文件名	说明	大小
struts2-quickvalidation-plugin-1.0-src.jar	Struts 2 QuickFormValidation	26.8

	1.0 sources 源码 Featured	KB
struts2-quickvalidation-plugin-1.0.jar	Struts 2 QuickFormValidation 1.0 jar Featured	26.8 KB
struts2-quickvalidation-demo.war	演示war包,部署至Tomcat6 运行 Featured	3.4 MB
Javascript_RapidValidation.pdf	JavaScript Rapid Validation 框架的详细使用及规则说明文档(转载)	176KB

5 演示应用

可下载示例应用来体验 Struts 极速验证框架，下载地址为:
http://struts-plus.googlecode.com/files/struts2-quickvalidation-demo.war ．放入Tomcat后即可查看运行效果．您可以本demo应用为基础快速的进行开发.

本演示包含三个内容:
纯后台表单验证
前后台一致表单验证(前台验证基于 RaplidValidation)
前后台一致表单验证(Struts 2 Form 标签)

本 war 包中的唯一的.java 文件源码如下:

```
package actions;

import com.opensymphony.xwork2.ActionSupport;
import common.validation.FormValidator;

public class HelloWorldAction extends ActionSupport {
// 或者implements Action {
    @FormValidator(input="form", value="user.name,用户名,required
min-length-5 max-length-10 validate-alpha;" +
           "age,年龄,validate-digits min-value-18 max-value-100;" +
           "password,密码,required;" +
           "password2,重复密码,required equals-password-原始密码;" +
           "email,,validate-email;" +
           "gender,性别,validate-one-required;" +
           "ip,,validate-ip;" +
           "date,生日,validate-date;" +
           "num1,数字1,validate-number;" +
           "num2,数字2,validate-number less-than-num1-数字1;" +
           "float,数字范围,float-range-1-10;" +
           "address,地址,validate-chinese;" +
```

```
            "zip,邮编,validate-zip;" +
            "id_num,身份证,validate-id")
    public String form() throws Exception {
        return SUCCESS;
    }

    @FormValidator(input="form_js", value="user.name,用户
名,required min-length-5 max-length-10 validate-alpha;" +
            "age,年龄,validate-digits min-value-18 max-value-100;" +
            "password,密码,required;" +
            "password2,重复密码,required equals-password-原始密码;" +
            "email,,validate-email;" +
            "gender,性别,validate-one-required;" +
            "ip,,validate-ip;" +
            "date,生日,validate-date;" +
            "num1,数字1,validate-number;" +
            "num2,数字2,validate-number less-than-num1-数字1;" +
            "float,数字范围,float-range-1-10;" +
            "address,地址,validate-chinese;" +
            "zip,邮编,validate-zip;" +
            "id_num,身份证,validate-id")
    public String form_js() throws Exception {
        return SUCCESS;
    }

    @FormValidator(input="form_tag", value="user.name,用户
名,required min-length-5 max-length-10 validate-alpha;" +
            "age,年龄,validate-digits min-value-18 max-value-100;")
    public String form_tag() throws Exception {
        return SUCCESS;
    }
}
```

下面是三个演示的效果:

Struts2QuickValidation后台验证测试页面 - 普通表单

用户名的长度不能小于5, 当前长度为3
用户名只能输入英文字母
年龄的值不能小于18, 当前值为17
密码不能为空
重复密码不能为空
email只能输入有效的电子邮件地址, 例如a@b.com
性别必须至少选中一项
生日的输入值不是有效的日期,正确格式为:yyyy-MM-DD
数字2的输入值1必须小于数字1的值1

用户名(user.name): 刘长炯 用户名的长度不能小于5, 当前长度为3
年龄(age): 17
密码:
密码重复:
邮件: a@b
性别: 男 女
IP地址:
数字1: 1 数字2(小于数字1): 1
生日: 1980-a-1
浮点数字范围(1~10):
住址:
邮编:
身份证:
提交

QuickValidation前后台统一验证演示页面

前台验证的更多信息请访问开源框架: http://code.google.com/p/rapid-validation/
提示:测试后台验证规则请禁用浏览器脚本功能!
普通表单和出错信息提示:
用户名不能为空
密码不能为空
重复密码不能为空
email只能输入有效的电子邮件地址, 例如a@b.com
性别必须至少选中一项

用户名(user.name): 请输入值.
年龄(age):
密码: a
密码重复: b 两次输入不一致,请重新输入
邮件: a@b 请输入有效的邮件地址,如 username@example.com.
性别: 男 女
IP地址:
数字1: 数字2(小于数字1):
生日:
浮点数字范围(1~10):
住址:
邮编:
身份证:
提交

QuickValidation后台验证演示页面(Struts 2 标签)

普通表单和出错信息提示: 用户名不能为空
年龄只能输入有效的阿拉伯数字,不含小数点等

- 用户名不能为空
  年龄只能输入有效的阿拉伯数字,不含小数点等

用户名:
年龄只能输入有效的阿拉伯数字,不含小数点等
年龄:
Submit

6 整合步骤 step by step

1. 将文件 **struts2-quickvalidation-plugin-1.0.jar** 放入 Web 项目的 **WEB-INF/lib** 目录下;
2. 修改您自己的代码的 **src/struts.xml** 中的 package 定义继承自 **quickValidation**;
示例代码:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
    "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
    <package name="test" namespace="" extends="quickValidation">
        <action name="helloworld" class="actions.HelloWorldAction">
            <result name="index">/index.jsp</result>
            <result name="success">/helloworld.jsp</result>
        </action>
    </package>

</struts>
```

3. 使用注解加入验证规则表达式;
在每个需要调用的 Action 的入口方法处, 加入**@FormValidator** 注解.
注解的完整语法如下:

```
@common.validation.FormValidator(input="出错时需要返回的result别名",
enabled=true/false, value="表单参数名称1,表单参数中文提示信息1,规则1 规则2…;
表单参数名称2,表单参数中文提示信息2,规则1 规则2…;更多字段验证信息")
```

input参数: 可选. 如果不指定, 则出错时仍然继续进入到Action方法中; 如果指定, 则在表单验证出错时, 返回到给定的result中, 例如: 原始表单输入页面或者统一的错误输出页面;
enabled参数: 可选. 是否启用此验证规则, 如果不指定, 则忽略解析注解, 不进行任何验证;
value参数: 必填选项, 可输入一个或者多个表单参数验证规则. 简单说就是: [字段1验证配置];[字段2验证配置];…. 虽然不建议, 然而字段验证配置可以留空, 此时不会进行任何验证处理.

而每个**验证配置**的格式如下:
表单参数名称: 必填选项. 指向输入的表单控件的名称(input 的 name);
表单参数中文提示信息: 可选. 如果为空, 则和表单参数名称相同. 不为空时, 则显示在出错时的信息提示的开头位置;
规则 1 规则 2…: 可选. 理论上可加入任意多个验证规则, 以**单个空格**隔开. 在验证开始后, 依次进行每个规则的验证并加入提示信息. 需要注意的是, 除 required 规则之外的所有规则只有当用户输入的字段值非空时, 才进行有效性验证, 如此便可避免所有字段都成为必填字段的问题发生.

每个规则本身则由两部分组成: 规则名-规则子名-参数 1-参数 2-参数 3-….

规则名和规则子名是固定名称，不能任意填写，这两个名称对应一个具体的规则实现类.
其中**参数**是**必填选项**，参数的意义则需要参考下一节的 验证规则表达式中的说明. 下面举例说明若干验证规则:

required 这是特殊规则，不带规则子名，也没有额外的参数;
float-range-1-20 其中 float-range 分别是规则名和规则子名，而 1 和 20 则分别是两个参数.

特别需要注意的是有负数时的情况此时参数列表变为: float-range--1--20，此时参数值分别为-1 和-20.

如果需要给另一字段加入验证规则，则以**分号**隔开.

下面是若干验证的示例及其解释:

规则	**解释**
@FormValidator	空的验证规则，不进行任何操作
@FormValidator("username, 用户名 ,required min-length-5")	对 username 字段进行验证，规则为必填，最小长度为 5; 出错时分别提示: 用户名不能为空 用户名的长度不能小于 5，当前长度为 1. 由于没有指定 input 参数，所以当验证失败时，仍然会继续进入到业务逻辑处理，相关的错误信息则存入 request 的 attribute 中.
@FormValidator("username, ,required min-length-5")	由于没有指定字段的中文提示，因此出错时将提示: username 的长度不能小于 5，当前长度为 1.
@FormValidator(**input="index"**,value=" username,用户名,required min-length-5”)	这大概是最常用格式的验证规则的写法了. 此例和上例的区别为: 所以当验证失败时，将进入 input 参数所指定的出错信息显示页面.
@FormValidator(input="index",**enabled=false,**value=" username,用户名,required min-length-5”)	忽略所有验证规则处理，常用于调试阶段.
@FormValidator(input="index",value=" username,用户名,required min-length-5;password2,**重复密码**,required equals-password-原始密码;”)	此规则加入了对重复密码字段的验证，出错时将提示: **重复密码**和原始密码的输入不一致
@FormValidator(input="index",value="username, 用户名,required min-length-5;password2,**重复密码**,required equals-password;”)	此规则省略了另一字段的中文描述，出错时将提示: **重复密码**和前面的输入不一致

以下为一个完整的 Action 类示例代码:

```
import com.opensymphony.xwork2.ActionSupport;
import common.validation.FormValidator;

public class HelloWorldAction extends ActionSupport {
    @FormValidator(input="index", value="username,用户名,required
min-length-5 max-length-10 validate-alpha;" +
            "age,年龄,validate-digits min-value-18
max-value-100;password,密码,required;" +
            "password2,重复密码,required equals-password-原始密码;" +
            "gender,性别, validate-one-required")
    public String execute() throws Exception {
        return SUCCESS;
    }
}
```

4. 修改前台显示验证结果.

根据 Action 的不同，前台的显示划分为两类:
普通的错误显示:

```
所有错误信息:  ${validationErrors}
```

返回的内容是一个字符串，显示了所有的出错信息，而出错字体的颜色则需要用户自己去设置.
出错信息的显示格式是以
分开的，例如:

```
用户名不能为空<br>密码不能为空<br>重复密码不能为空<br>性别必须至少选中一项<br>
```

另一种推荐的做法是让 Action 继承自 Struts2 中的 ActionSupport，此时则可用 Struts 2 的标签来显示出错信息，可定位到具体的字段:

```
<%@ taglib prefix="s" uri="/struts-tags" %>
<s:actionerror />
<s:fielderror/>

  <s:form action="helloworld.action" method="post">
   <s:textfield name="username" label="用户名" />
   <s:password name="password" label="密码" />
   <s:submit/>
   </s:form>
```

或者使用纯 JSP 来编写页面来定位到单条错误:

```
用 户 名 : <input name=username value=${param.username}>
${errors.username[0]}
```

下面给出这两种写法的完整示例:

```
<%@ page pageEncoding="UTF-8"%>
```

```
<%@ taglib prefix="s" uri="/struts-tags" %>
<html>
  <head>
<style>
.errorMessage {color:red;}
.errorLabel {font-weight:bold}
</style>
    <title>测试页面</title>
  </head>
  <body>

<s:actionerror />

   <form action="helloworld.action" method=post
enctype="multipart/form-data">
   用户名: <input name=username value=${param.username}>
${errors.username[0]} <br>
   年龄: <input name=age value=${param.age}><br>
   密码: <input name=password><br>
   密码重复: <input name=password2><br>
   邮件: <input name=email><br>

   性别: <input type=checkbox name=a>男 <input type=checkbox name=a>女<br>
   IP地址: <input name=ip><br>
   数字1: <input name=num1> 数字2: <input name=num2> <br>
   日期:<input name=date><br>
   附件: <input name=file type=file><br>
   <input type=submit value="提交">
   </form>

<span class="errorMessage">
${validationErrors}
</span>

Struts 2表单标签示例:

<s:fielderror/>

  <s:form action="helloworld.action" method="post">
   <s:textfield name="username" label="用户名" />
   <s:password name="password" label="密码" />
   <s:submit/>
   </s:form>
  </body>
```

```
</html>
```

7 验证规则表达式详解

说明: 加红色背景的部分是目前尚不支持的验证规则; 加黄色背景的是增加了第二个可选参数的规则(和前台验证规则略有变动).

验证表达式	描述	示例
required	非空域,全部空格也算空	
validate-number	一个有效数	
validate-digits	只能包含 0-9 任意个数字	
validate-alpha	只能是字母 a-z, A-Z	
validate-alphanum	只能是字母和数字的组合	
validate-email	只能是有效的 email	
validate-url	只能是有效的 url 地址	
validate-one-required	至少有一个被选中,例如一组 checkbox, radiobutton,它们最好包含在一个 div 和 span 中	
validate-integer	只能是整数,可以有正负号	
validate-ip	有效的 IP 地址	
min-length-$number	最小长度是$number **(此处 $some 表示一个特定的值)**	最小长度为 8: min-length-8
max-length-$number	最大长度是$number	最大长度为 8: max-length-8
max-value-$number	输入域的最大值是$number	最大值为 8.1: max-value-8.1
min-value-$number	输入域的最小值是$number	最大值为-8.1: max-value--8.1
equals-$otherInputId-$otherInputLabel	必须和某个 input field 相等,用于密码两次输入验证等,$otherInputLabel 是可选项,用来描述另一字段信息	equals-password 重复密码和原始密码输入不一致: equals-password-原始密码
less-than-$otherInputId-$otherInputLabel	小于某个 input field less-than-otherInputId,多用于结束日期不能小于开始日期的需 求,$otherInputLabel 是可	注:相等时也不能通过

	选项,用来描述另一字段信息	
great-than-$otherInputId-$otherInputLabel	大于某个 input field less-than-otherInputId,$otherInputLabel 是可选项,用来描述另一字段信息	
validate-date-$dateFormat	只能是日期,$dateFormat 为日期格式,$dateFormat 的可选,默认格式为 yyyy-MM-DD	validate-date-yyyy 年 MM 月 dd 日
validate-file-$type1-$type2-$typeX	验证文件输入域选择的文件类型只能为声明的$type1 - $typeX 中的一种	validate-file-png-jpeg
float-range-$minValue-$maxValue	必须是$minValue 到 $maxValue 的一个浮点数	1 至 20: float-range-1-20
int-range-$minValue-$maxValue	必须是$minValue 到 $maxValue 的一个整数	
length-range-$minLength-$maxLength	输入字符串的长度必须在 $minLength 到$maxLength 之间	
validate-pattern-$RegExp	通过自定义正则表达式 $RegExp 来验证输入域的正确性	vaidate-patter n-/a/gi
validate-chinese	只能是中文(以下为中国的相关验证)	
validate-phone	有效的电话	
validate-mobile-phone	有效的手机号	
validate-id-number	验证是否有效的身份证号码	
validate-zip	验证邮政编码	
validate-qq	验证 QQ 号码	

8 和 RapidValidation 配合实现前后台一致验证

RapidValidation (http://code.google.com/p/rapid-validation/) 是专注于前台的验证框架, 例如下面的例子:

```
<!-- 为 form 增加 required-validate class,标识需要验证 form -->
<form id='helloworld' action="#" class='required-validate'>
    helloworld:</br>
    <!--通过 class 添加验证: required 表示不能为空,min-length-15
```

```
    表示最小长度为 15 -->
        <textarea name='content' class='required
    min-length-15'></textarea></br>
        <input type='submit' value='Submit'/>
        <input type='reset' value='Reset'/>
</form>
```

其对应的后台验证规则为:

```
@FormValidator(input="index",value=" content,helloworld,required min-length-15")
```

请注意比较异同, 不同部分已经以红色粗体进行了区分, 相同部分则以绿色粗体标识.
大家可以看到, 两者很容易就实现了一致的前后台验证规则.

相关的说明文档请访问: http://code.google.com/p/rapid-framework/wiki/rapid_validation , 另外本项目也提供了一份完整文档的转载, 包含详细的规则说明文档和整合步骤: (http://struts-plus.googlecode.com/files/Javascript_RapidValidation.pdf).

9 如何扩展验证规则

要扩展验证只需要实现 **IValidator** 即可即可, 并将实现类放入包 **common.quickvalid.validators** 下. 下面是 **IValidator** 接口定义:

```
package common.quickvalid.validators;

import java.util.Map;

/**
 * 所有验证器的抽象接口.
 * @author beansoft
 *  日期: 2009-12-20
 */
public interface IValidator {
    /**
     * 实现验证方法.
     * @param fieldName 字段name
     * @param fieldDescription 出错时的字段描述信息
     * @param params 参数列表 <String key, String[] values> 表单参数 Map
     * @param args 参数列表
     * @return 验证信息, 为空时表示没有任何出错信息, 通过验证
     */
    public String doValidation(String fieldName, String
fieldDescription, Map<String, String[]> params, String... args) ;

    /**
     * 获取验证器本身的参数列表.
```

```
     * @return
     */
    public String[] getArgs();

    /**
     * 设置验证器本身的参数列表
     * @param args - 一个或者多个参数, 允许为 null
     */
    public void setArgs(String[] args);

    /**
     * 获取原始表达式.
     * @return
     */
    public String getExpression();

        /**
     * 设置原始表达式, 用于日期, 正则等验证方式, 避免和 - 号冲突.
     * @param Expression
     */
    public void setExpression(String expression);
}
```

在实践中，已经提供了几种比较常见的实现类的抽象基类，它们是:
BaseValidator，抽象的验证器实现，仅实现了 setter 和 getter;
BaseRegexValidator(继承自 BaseValidator)，基于正则表达式验证的基类;
BaseTwoNumberValidator(继承自 BaseValidator)，提供基于两个数字规则的验证，可获取两个参数(含-号).

类名有一定规则，将第一个减号替换为下划线即可，例如:
float-range-$minLength-$maxLength 对应的类名是：*length_range*.

下面则给出一些实现类，供参考:
手机号的验证:

```
package common.quickvalid.validators;

/**
 * 手机号验证
 * TODO 更多号段的支持?
 * @author beansoft
 *
 */
public class validate_mobile extends BaseRegexValidator {
```

```
    @Override
    public String getMessage() {
        return "只能输入正确的手机号码,如:13810390000";
    }

    @Override
    public String getRegEx() {
        return "(^0?[1][35][0-9]{9}$)";
    }
}
```

数字范围的验证:

```
package common.quickvalid.validators;

import java.util.Map;

/**
 * 表单域取值浮点数验证
 * @author beansoft
 *
 */
public class float_range extends BaseTwoNumberValidator {

    public String doValidation(String fieldName, String
fieldDescription,
            Map<String, String[]> params, String... args) {
        super.setParams(params);
        String value = getValue(fieldName);
        args = super.parseTwoNumberArgs();//重新分析

        // 非空时进行验证
        if(value != null && value.length() > 0) {
            boolean result = false;
            try {
                result =  (Double.parseDouble(value) <
Double.parseDouble(args[0])) ||
                    (Double.parseDouble(value) >
Double.parseDouble(args[1]));
            } catch(Exception ex) {
                return fieldDescription + "的输入值不是有效的数字";
            }

            if(result) {
```

```
                return fieldDescription + "的输入值" + value + "应该为" +
args[0] + "到" + args[1] + "之间的数字";
            }
        }
        return null;
    }

}
```

10 下一版本开发计划

- 将验证规则表达式解析和执行部分独立出来作为核心引擎
- 兼容 JDK 1.4 并提供独立调用的接口
- 提供 Struts 1 下的插件机制
- 提示文字国际化
- 增强前台验证，提供时分秒的验证功能

如果您对本项目有兴趣，欢迎加入.